begindocument/before

Low-cost, open source, analog polysynth based on the AS3397 chip

D. Guichaoua[1, a)] and M. Loumaigne[1, b)]

Laboratoire MOLTECH-Anjou, UMR CNRS 6200, Université d'Angers, SFR MATRIX, 2 Bd Lavoisier, 49045 Angers CEDEX, France

We present a low-cost, open-source, polyphonic synthesizer based on the AS3397 analog synth voice chip. Utilizing the versatile RP2040 microcontroller for each independent voice, our architecture supports a scalable polyphony up to virtually more than 100 voices. A central "conductor board," also driven by an RP2040, orchestrates the voices, offering unprecedented control and flexibility. Our paper provides comprehensive documentation of both the hardware and software aspects, aiming to facilitate replication and modification of the design. The thorough explanation of each component and the software stack serves as a valuable resource for further academic and applied endeavors in the field of electronic musical instruments : we have created this synthesiser with the intention of inspiring the open-source community to embrace, modify, and enrich this project.

I. INTRODUCTION

In today's musical landscape, digital synthesis has reached a point where it can faithfully replicate the capabilities of analog synthesizers[?] . Technically speaking, limitations concerning aliasing and latency have been overcome for over a decade now. Despite these advancements, there remains a sustained interest in analog synthesis. This is evidenced by the frequent modern reissues of classic synthesizers from the 1970s and 1980s, as well as the elevated prices of vintage models on the second-hand market.

From a sonic and rational perspective, there may be limited reasons to opt for analog synthesis in 2023. What remains are mainly aesthetic considerations: the allure of introducing an element of randomness, a fondness for the quirks and imperfections, and, maybe, a desire for tangibility in an increasingly intangible world. Additionally, from an educational standpoint, working with analog synthesizers provides an invaluable experience in learning electronics, printed circuit board (PCB) design, and microcontroller programming.

However, the construction of a polyphonic synthesizer is a colossal task, involving approximately a thousand electrical components and complex logic circuits to control them. Remarkably few, if any, projects of this scale exist within the open-source community. Commercially available polyphonic synthesizers of this type often come with a price tag running into several thousands of euros.

Building a stable and reliable polyphonic synthesizer with discrete components is a monumental challenge. Each voice of polyphony requires hundreds of components, adding complexity to assembly, elevating the risk of wiring errors, reducing reparability, and significantly expanding the circuit's footprint. Furthermore, components like transistors must be carefully matched, and thermal drift among the components can affect the synthesizer's sonic stability. This set of challenges makes a compelling case for the need for a dedicated chip to handle the intricacies of polyphonic synthesis.

a)Electronic mail: dominique.gichaoua@univ-angers.fr
b)Electronic mail: matthieu.loumaigne@univ-angers.fr

In this article, we present an open-source analog and polyphonic synthesizer designed with several guiding principles in mind. The architecture is easy to understand, modify, and repair, while also being cost-effective and educational. This holistic approach aims to offer a practical and accessible solution in the realm of analog and polyphonic synthesis, with the intention of inspiring the open-source community to embrace, modify, and enrich this project.

In this work, the sonic architecture revolves around the AS3397 chip developed by the Latvian semiconductor manufacturer ALFA RPAR. This chip is a modern reissue of the CEM3396 chip originally developed by Curtis Electromusic in May 1984. The CEM3396 can be found in classic synthesizers like the Oberheim Matrix-6 and, in a slightly modified form, in the Prophet 08.

The structure of this article is organized as follows:

- First, we delve into the overall design architecture, with a particular focus on managing polyphony. We describe the role of a "conductor" board that controls multiple "voice" boards, each of which serves as an independent monophonic synthesizer. We also present the architecture of an individual voice, detailing the sonic capabilities of the AS3397 chip and its implementation within our system.

- After outlining the general architecture, we briefly discuss some of the key technical challenges we encountered and how we successfully addressed them.

- Lastly, while we recognize that the aesthetic quality of the sound synthesis may be subjective, we provide measurement results obtained from the synthesizer and discuss their alignment with the design specifications we initially set.

Before diving into the main text, it's worth noting that a particularly detailed Electronic Supplementary Information (ESI) accompanies this article. The ESI is designed to function as a tutorial, covering topics that may be straightforward for experts but could potentially be stumbling blocks for beginners.

II. OVERALL DESIGN ARCHITECTURE

A. Overall architecture of the polyphonic synthetiser

The overall architecture of the polyphonic synthesizer is shown in fig. 1.

The conductor board, built upon the RP2040 microcontroller and more precisely using the pico-W prototyping board, receives musical input in the MIDI format and dispatches it to each of the voices using the i²c protocol. The analog sound signal is generated by AS3397 chips using mainly subtractive syntheses. The stereo audio signals produced by every voice are subsequently mixed together. We've implemented an effects loop utilizing the second core of the RP2040. The outgoing analog signal is digitized by a codec (Coolaudio 4220) and conveyed to the RP2040 via i²S. After processing, the signal is converted back to analog using the same codec. The raw analog signal and the FX signal are then blended to produce the synthesizer's output.

B. Architecture of a single voice

1. Overview of the AS3397 chip

Figure **??** shows the structure of the AS3397 chip.

The Voltage Control Oscillators (VCO) employed in the AS3397 chip are of the Digital Controlled Oscillator (DCO) variety. A constant current source consistently charges the CTA capacitor. This capacitor can be swiftly discharged by grounding it via a transistor. This switch is activated by a square wave voltage sent to the transistor's base. The voltage, dictating the charge/discharge cycle frequency of the capacitor and, ultimately, the frequency of the note produced by the DCO, is generated through a Pulse Width Modulation (PWM) output of the RP2040 (refer to SI– for additional details).

Contrasting other analog VCOs, frequency control by a microcontroller, and subsequently by quartz oscillations, ensures remarkable frequency stability and significantly simplified tuning. As will be discussed in Section 3, a DCO still requires calibration. We opted to directly measure the voltage ramp at the capacitor's terminals using one of the RP2040's ADCs, controlling the current generator's voltage with a PID correction.

The voltage ramp at the capacitor's terminals is sent to a waveform shaper, which outputs waveforms ranging from a ramp to a clipped triangle, through a triangle. Concurrently, the capacitor's voltage is also directed to a comparator, whose output yields square waves with a duty cycle controlled by the comparison voltage. A multiplexer (referred to "Quad Level Driver" in the datasheet) allows the user to decide if they wish to sum the waveform shaper's outputs (ramp/triangle) with the square waveforms from the comparator.

The AS3397 features two DCOs, the signals of which can be mixed using two VCAs controlled by the Balance C.V. input. Before reaching the VCF section, an external signal can be passively mixed with the outputs of the two DCOs. Here, we use this input to mix a DSO (Digital Signal Oscillator) generated by the RP2040. The VCF is a resonant 4-pole low-pass filter, schematically represented as a sequence of four first-order low-pass filters sharing the same pole. The filter's cutoff frequency ($\omega_c = 1/(RC)$) is achieved using a fixed-value external capacitor to the chip ($C = 22\,\text{nF}$), but with a variable equivalent resistance R that's voltage-controlled via an OTA (Operational Transconductance Amplifier) (see SI–X).

The signal exiting the VCF is DC-coupled to a primary VCA determining the general gain and two VCAs managing the stereo balance.

Figure **??** illustrates how, on a voice card, the RP2040 microcontroller drives the AS3397 chip. The microcontroller generates the 10 voltages required to control the inputs of the AS3397. It also produces two square signals that determine the frequencies of the two DCOs. This frequency is adjusted through a PID mechanism, based on digitizing the voltage across the charging capacitors' ramps. Furthermore, to enrich the synthesizer's sonic palette, a digitally generated audio signal is produced by the RP2040 to be mixed with the AS3397's analog DCO signals.

III. METHODS

A. Hardware

1. Microcontroller : RP2040

Our specifications for the microcontroller responsible for controlling the AS3397 chip required it to be: sufficiently fast to produce high-frequency PWM outputs with adequate resolution (as discussed in Section III A 4); well-documented with a significant user community; equipped with a suitable number of General Purpose Input Output (GPIO) pins to manage all the inputs of the AS3397 chip; cost-effective; energy-efficient; and readily available during this electronic chip shortage period.

The need for a sufficiently high frequency ruled out the venerable ATMEGA328p, commonly used on the Arduino platform, which operates at 16 MHz.

Introduced in January 2021 by the Raspberry Pi Foundation, the RP2040 has gained substantial traction two years post its release. The plethora of online resources and dedicated libraries make it a well-regarded choice. The microcontroller boasts dual ARM Cortex-M0+ cores, typically clocked at 133 MHz—though it's worth noting that overclocking them up to 250 MHz is straightforward. With a cost hovering around 1 euro, a consumption rate of approximately 30 mA, 26 GPIO pins, and its current easy availability, the RP2040 presents an appealing option. Notable limitations include its RAM capacity of just 264 kB, which, fortunately, poses no issues for our project.

While designing an electronic circuit to implement an RP2040 is well-documented, we opted for the development board pi-pico, indeed, manually soldering the RP2040 chip, which comes in a QFN package, can be challenging for novices.

2. Power supply

Each voice needs three power voltage, namely 5 V for the rp2040 and the command voltage of the AS3397, 12 V and -5 V for the AS3397. La carte maitre accepte une tension non stabilisée de 15V et créé trois rail de Tension via des LDO. Chacune de ces tensions sont envoyés aux cartes voies. Le buck converter des cartes pi pico créé alors le rail 3.3 V à partir du 5 V. Une carte voie apelle un courant de 10 mA, 20 mA et 30 mA sur les rails -5 V, 5 V et 12 V respectivement.

The master board accepts an unregulated 15 V input and creates three voltage rails via Low Dropout Regulators (LDOs). Each of these voltages is distributed to the voice cards. The buck converter on the Pi Pico boards then generates the 3.3 V rail from the 5 V supply. A single voice card draws current of 10 mA, 20 mA, and 30 mA from the -5 V, 5 V, and 12 V rails, respectively.

3. Communication via i²c

We elected to utilize the i2c protocol, operating at a bus frequency of 400 kHz, to facilitate communication between the conductor board (master) and the rp2040 on the voice boards (slaves). The rationale behind this choice was its straightforward implementation, requiring only two wires and two pins on the rp2040. The bi-directional bus nature of i2c also allows the conductor board to receive feedback from the voice boards, particularly regarding the availability of voices, which aids in polyphony management. As a side note, the use of the broadcast mode that dispatches identical information to all voice boards helps conserve bandwidth.

On the I²C bus, the 8-bit messages emulate the protocol used by MIDI (Musical Instrument Digital Interface). This covers the management of notes (both 'note on' and 'note off'), as well as velocity, aftertouch, control changes (MIDI CC), and other parameters. Additionally, extended MIDI is incorporated to overcome the standard MIDI's limitation of 127 levels when necessary.

The addressing of each voice board is automatically carried out upon the startup of the conductor board. More specifically, a voltage—derived by dividing the 3.3V generated by the conductor board and uniquely distributed to each voice board—is read by the third ADC of the voice board's rp2040. This informs it of the specific i2c channel allocated to it. At the startup of the conductor board, it scans all the utilized i2c ports, enabling it to determine the synthesizer's total polyphony.

4. Filtered Pulse Width Modulation as Digital to Analog Converters

The AS3397 chip features 10 voltage-controlled inputs that need to be driven by a voltage generated by the RP2040 to eventually manage polyphony.

Initially, one might consider employing 10 Digital to Analog Converters (DAC). However, firstly the RP2040 lacks an integrated DAC, and generally, DACs built into microcontrollers don't deliver high performance. An alternative would be to utilize a dedicated 8-channel DAC, such as the MCP3208. Yet, the associated chips for such solutions tend to be relatively expensive, often costing several euros.

A middle-ground solution is to use a single-channel DAC in tandem with a multiplexer. This setup enables sequential and rapid addressing of all the AS3397 inputs while synchronously adjusting the DAC. The AS3397 was designed with this mode of operation in mind, given its high-impedance inputs. By paralleling an input with a capacitor, the voltage can be retained as the multiplexer switches to the next AS3397 input.

However, this approach amplifies the component count, overall cost, and software complexity. Thus, driven largely by a preference for efficiency and simplicity, we decided to harness the RP2040's PWM outputs, filtering them to retain only the direct current component, effectively replicating the functionality of a DAC (see section SI–12 for detailed mathematical and technical aspects).

In brief, we chose a PWM signal frequency of approximately 32 kHz for most controls due to two primary reasons:

- Given the microcontroller's frequency (130 MHz), this allows for 4096 distinct duty cycle levels, equating to a 12-bit resolution. This granularity is more than adequate for most synthesizer controls. One must understand that this choice isn't dictated by electronic considerations but rather physiological ones. Beyond a certain precision, our ears can't discern the differences. For comparison, the long-standing standard MIDI encodes control values over only 127 levels (7 bits).
- The modulation frequency of the PWM falls outside the audible range. Even if it's not entirely inaudible, it won't be perceptible. Still, one should be cautious of inter-modulation phenomena that might introduce frequencies within the audible domain. Consequently, a simple RC first-order low pass filter is sufficient to keep the DC value of PWM signals.

The low-pass filter's cut-off frequency f_c, hence the determination of R and C values, results from a trade-off between:

- The suppression of the PWM signal's carrier and its harmonics. Thus, we desire the lowest possible cutoff frequency $f_c = 1/(2\pi RC)$. If the signal isn't filtered adequately, a residual ripple superimposes on the desired average signal value.
- The filter's response time, which is roughly $10 \times RC$. This time has to remain in line, with our psychoacoustic limitations, typically around 120 ms**?** .

We opted to use a response time of 6 ms corresponding to a residual oscillation of ± 10 mV. For the VCF's cutoff frequency control, to which our ears are notably sensitive, we prioritized lower residual oscillation (± 5 mV) over response time (14 ms).

Furthermore, certain control voltages, particularly those governing waveform shape (see section **??**), require greater precision than 4096 levels. We employed the signal from the sum of two 8-bit PWMs to craft a single 16-bit control signal using a passive summator. This summator combines V_{low}, encoding the 8 least significant bits, with V_{high}, encoding the 8 most significant bits, forming a unified 16-bit command. It's crucial that resistances R_{high} and R_{low} maintain a 256:1 ratio, and we opted for 1% precision resistors ($R_{high} = 3.9\,\mathrm{k\Omega}$ and $R_{low} = 1\,\mathrm{M\Omega}$). Furthermore, having only an 8-bit resolution permits elevating the PWM frequency to 500 kHz, ensuring better carrier suppression.

It's worth noting that the DSO signal generated by the RP2040 also derives from a filtered 16-bit PWM signal.

The PWM signals from the RP2040 have an amplitude of 3.3 V. However, the AS3397 chip, which in some ways harks back to designs from the 1980s, operates with voltages ranging from 0 to 5 V. Some inputs, such as those for cut-off or for controlling the duty cycle of the square wave portion of the signal, even require negative voltages to reach their lower limits. Consequently, the filtered PWM outputs are routed through level shifters: specific ICs (CD4050) are used to transition from 3.3 V to 5 V, while op-amp configurations are employed when a negative lower voltage boundary is needed.

5. *Calibration of the DCO via PID feedback*

A DCO allows for a precisely controlled oscillator frequency. However, the trade-off is the need to compensate for the ramp height, which varies with the frequency of the played note. For the AS3397, the maximum value V_{ramp} reached by the voltage ramp across the capacitor determines the waveform output from the waveshaper. We aim for this waveform to remain consistent regardless of the frequency of the synthesized note.

The maximum voltage V_{ramp} is determined by the slope of the charge and its duration. The slope, or the charge rate, is influenced by three parameters: the capacitance C of the capacitor, the resistance R, and the charging voltage of the set V_{wf}.

The charging duration T of the capacitor is dictated by the frequency f with $T = 1/f$.

If we maintain a constant V_{wf}, higher frequency notes have less time to charge the capacitor, resulting in a lower V_{ramp} compared to a lower frequency note under the same conditions. Hence, with a constant V_{wf}, the waveform output from the waveshaper will differ across frequencies. Consequently, we must adjust the charging voltage V_{wf} to ensure a consistent charging height, irrespective of the desired frequency f.

More specifically, the maximum value V_{ramp} of the ramp is given by:

$$V_{\text{ramp}} = \underbrace{\left(\frac{V_{wf}}{RC}\right)}_{\text{slope}} \underbrace{\frac{1}{f}}_{\text{duration}} \tag{1}$$

In practice, the time constant $\tau = RC$ is not well-defined because the measurements of R and C can be imprecise, and these values change with temperature and aging. One solution is to calibrate each voice of the synthesizer during the startup phase. We have opted to implement a feedback control system, where we measure the ramp height and then adjust the charging voltage V_{wf} to maintain a consistent ramp height V_{ramp} regardless of the oscillator frequency. More precisely, the capacitor discharge pulses are managed by an interrupt that is triggered at the desired frequency f for the note. During this interrupt, three operations are carried out:

1. The current value is read using the ADC. Since the next step of the interrupt is to discharge the capacitor, we can ascertain that the ramp value is at its peak during its ADC reading.
2. The capacitor is discharged (by sending a pulse to the discharge transistor gate).
3. The signal provided by the ADC is analyzed against the setpoint. A Proportional-Integral (PI) control algorithm is applied to determine the appropriate V_{wf} value, ensuring that the ramp quickly stabilizes without oscillations at the desired maximum value.

As recommended in the datasheet, we have selected a 1.5 nF capacitor for charging the VCO. Regarding the associated charge resistance, we followed the datasheet's strategy of using two resistance values: the first being $R_{\text{low}} = 640\,\mathrm{k\Omega}$ for lower frequencies where the capacitor needs to charge slowly, and the second $R_{\text{high}} = 20\,\mathrm{k\Omega}$ for situations where the capacitor must charge quickly. The transition point for switching between these two registers occurs at a frequency of $f_{\text{register}} = 512\,\mathrm{Hz}$. The register change is executed using a transistor to short the $R_{\text{high}} = 20\,\mathrm{k\Omega}$ resistance to ground.

6. *FX Loop*

We implemented the circuit as described in the application note found in the datasheet of the Coolaudio 4220 codec. For audio input and output data transfer, we utilized the I2S protocol, which is implemented on the RP2040 using its Programmable Input Output (PIO)

7. *PCB*

We utilized the software KiCad (v7.0) for circuit design and PCB routing.

Following a prototyping phase using a homemade printed circuit board with through-hole components, we routed a two-layer circuit using Surface-Mounted Device (SMD) components. Components were selected in the 1206 format to facilitate easy manual soldering and desoldering. The PCB fabrication and component assembly (with the exception of the AS3397 and the Pi Pico) were carried out by JLPCB. The conductor card was assembled on a protoboard and orchestrates 5 voice cards.

B. Software

1. *Programming language*

For code efficiency reasons, the rp2040 was programmed in C++ instead of MicroPython. Furthermore, to leverage the most prevalent platform among hobbyists, we employed the software layer of the rp2040 adapted to Arduino syntax and utilized the Arduino IDE version 2.1.

Architecture du code. Séparation entre les différents fichiers. Les interruptions. La lecture des inputs en I2C. La carte voie et la carte orchestre.

Commentaire sur le code.

C. Implementation of the Digital Signal Oscillator

We calculate and set the PWM output values for each call of an interrupt running at 65 536 Hz. These calculations are performed on the second core of the RP2040, leaving the first core to manage the control of the AS3397 chip. At the time of writing, we have implemented wavetable synthesis with a separate wavetable for each octave to minimize aliasing. The DSO also allows for the generation of white noise. It's important to note that the DSO, inherently digital, is also filtered by the analog VCF. This section is intentionally kept simple; other types of synthesis, like FM, could be implemented to complement and enrich the sound palette of the AS3397's analog subtractive synthesis.

D. FX

Delay, reverb, chorus.

Les calculs des effets sont fait

E. Total cost

IV. RESULTS

A. Tuning of the VCO amplitude via PID

As mentioned earlier, the use of a DCO ensures perfect frequency tuning of the oscillator. However, it's necessary to adjust the ramp height for each frequency of the DCO. Poor adjustment in this case, due to the presence of the waveshaper, will result in fluctuation of the waveform with frequency, leading to a slightly different timbre for each note.

The successive values of the maximum ramp voltage V_{ramp} for one of the VCO's capacitors were recorded directly by the RP2040's ADCs during the tuning interruptions. Figure **??** displays the evolution of the error relative to the target ramp voltage, the value of the residual ripple around the achieved voltage, and the settling time as a function of the frequency for the 127 MIDI notes. The register change at the frequency $f_{\text{register}} = 512\,\text{Hz}$ is clearly visible.

On average, the discrepancy between the target value and the achieved value is about x%, with an average settling time of $100\,\mu\text{s}$. Both phenomena are imperceptible to the ear.

B. Maximum i2c transfert rate

Décrochage des notes ? Faire un fréquency sweep avec les notes midi à des vitesses de plus en plus grande. Peux-ton avoir des modulations aux fréquences audio. Est-ce compatible avec les 400khZ.

C. Filtered PWM control voltage signal

Figure **??** displays the settling time and the residual oscillations of the control voltages (CV) obtained by filtering PWM signals with a carrier frequency of 32 kHz and a filtering frequency of $f_{RC} = 10\,\text{kHz}$. The residual fluctuations in the CVs are inaudible, and the response time, being below the physiological limit, is not perceptible.

Figure **??** illustrates the voltage response curve of the 12-bit DAC thus obtained.

D. VCF characterization

Réponse percu ou au bruit via le DSO. Mesure sur une carte son (à mettre dans material and methods).

Fermeture du filtre ? Auto-résonance. Vérifier la datasheet ?

E. DSO distortion

Mesure à 1 kHz d'une sinusoïde.

V. CONCLUSION

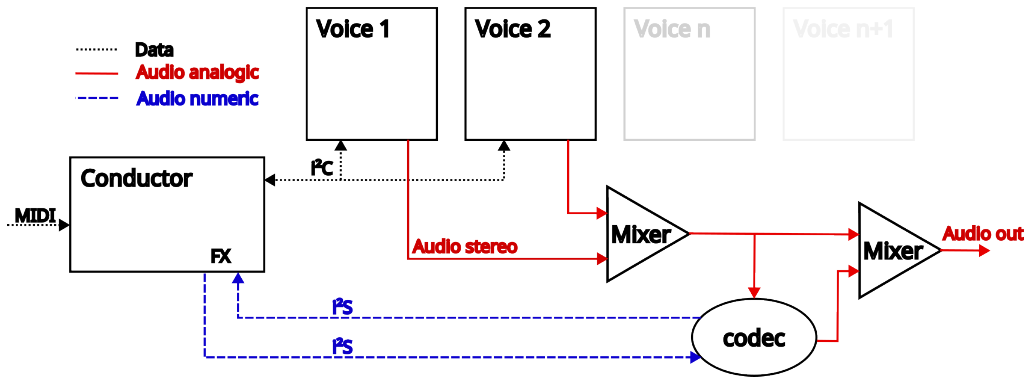

FIG. 1. Overall architecture of the polyphonic synthesizer. For clarity, the power bus lines are not shown.

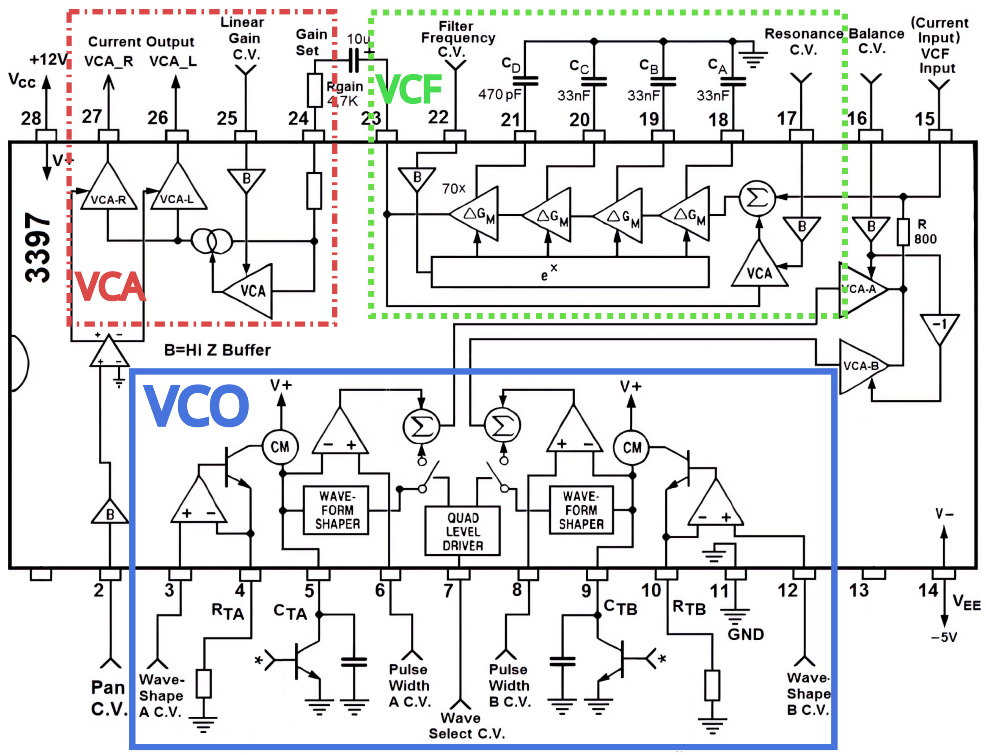

FIG. 2. Slightly simplified schematic block diagram of the AS3397 chip

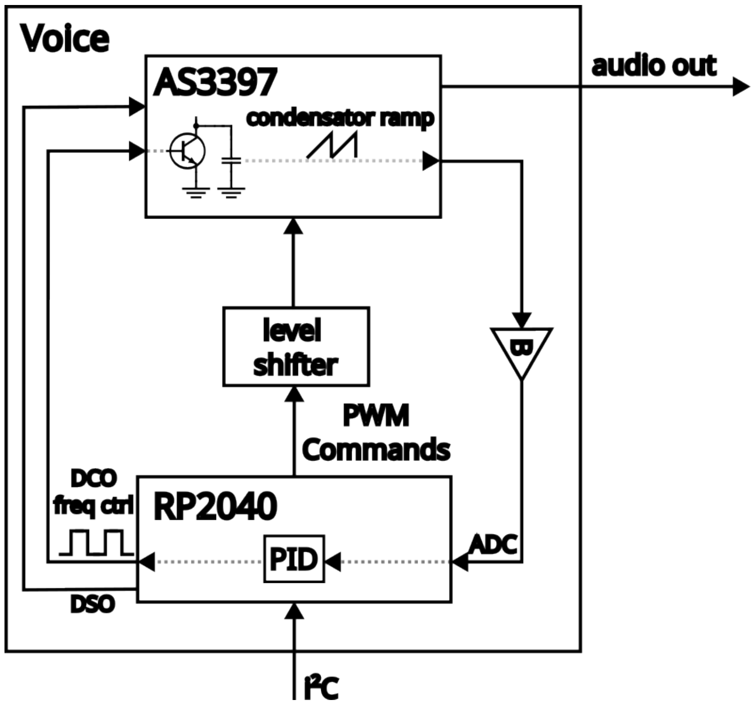

FIG. 3. Schematic block diagram of the control of the AS3397 chip by the RP2040 micro-controller.